МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

И ПРОГРАММНОЙ ИНЖЕНЕРИИ (КАФЕДРА №43)

|  |  |  |
| --- | --- | --- |
| ОТЧЕТ ЗАЩИЩЕН С ОЦЕНКОЙ: |  |  |

ПРЕПОДАВАТЕЛЬ:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Старший преподаватель |  |  |  | Е. В. Павлов |
| (должность, уч. степень, звание) |  | (подпись, дата) |  | (инициалы, фамилия) |

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №4

«ОЦЕНКА АЛГОРИТМИЧЕСКОЙ СЛОЖНОСТИ

ПРОГРАММНОГО КОДА»

ПО КУРСУ: «МЕТРОЛОГИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ»

|  |  |  |  |
| --- | --- | --- | --- |
| РАБОТУ ВЫПОЛНИЛ (-А) СТУДЕНТ (-КА): | 4931 | / | Е. Ю. Ильченко |
|  | (номер группы) |  | (инициалы, фамилия) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | / |  | / | 08.05.2021 |
|  |  | (подпись студента) |  | (дата отчета) |

1. Цель работы

Целью данной работы является анализ графа потока управления и оценка алгоритмической сложности программного кода на основе метрики Маккейба.

1. Задание на лабораторную работу

Начертить блок-схему алгоритма программного кода приложения (или его фрагмента), построить граф потока управления с использованием языка DOT, выделить линейно-независимые маршруты и циклы, выполнить расчёт цикломатического числа Маккейба.

Отразить в выводах результаты и проанализировать корректность расчета цикломатической сложности.

Разрешается ограничить исходный код программы (из ЛР 1) и использовать для построения блок-схемы алгоритма и графа потока управления фрагмент кода (или модуль) размером 120-160 строк (из которых не более 15% пустых). Данный фрагмент кода (или модуль) обязательно должен содержать как операторы ветвления, так и циклические конструкции.

|  |
| --- |
| Вариант задания:  6. Приложение для поиска туров (по стране) |

1. Оценка алгоритмической сложности программного кода

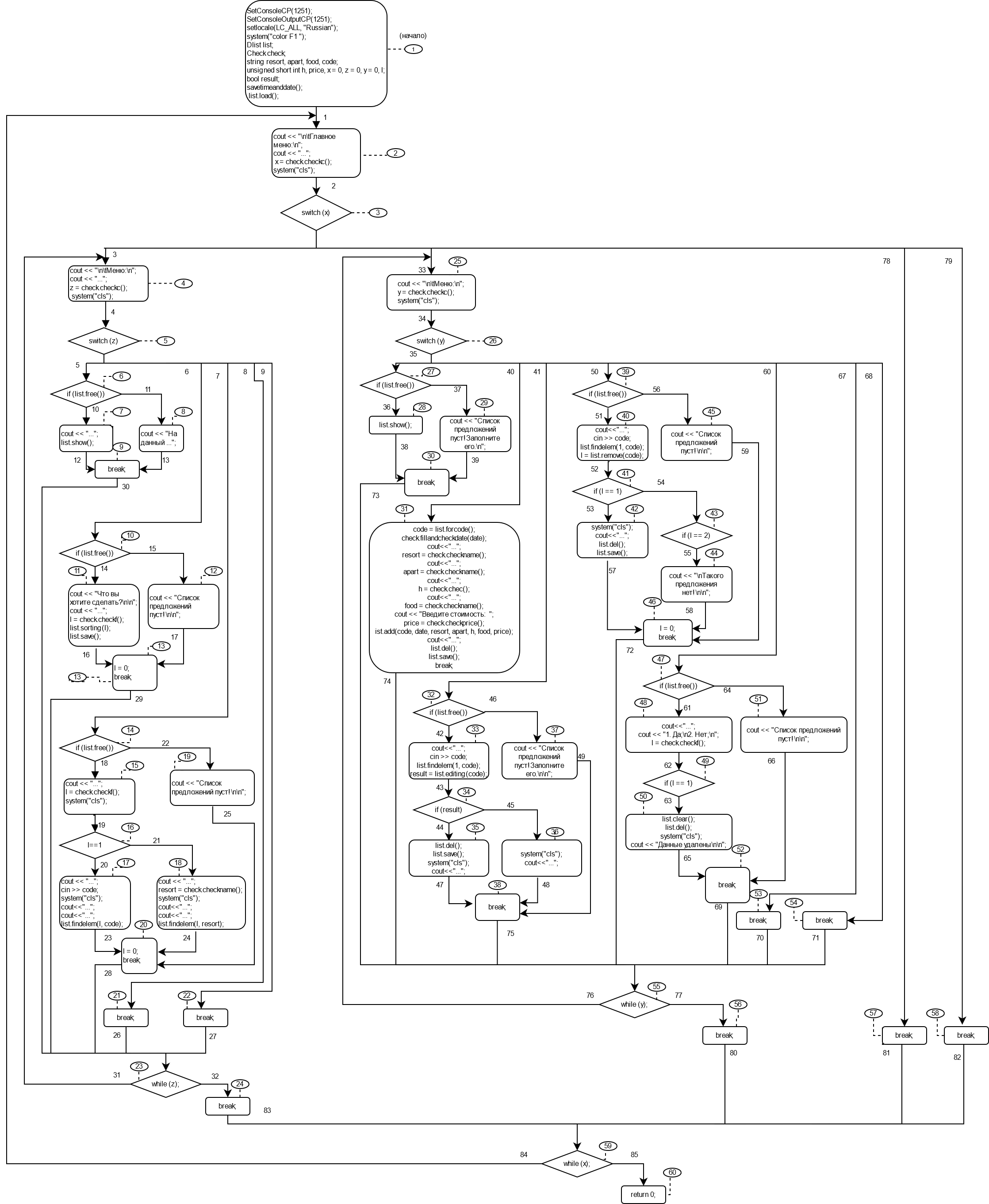


Рисунок 1 — Блок-схема алгоритма анализируемого фрагмента кода

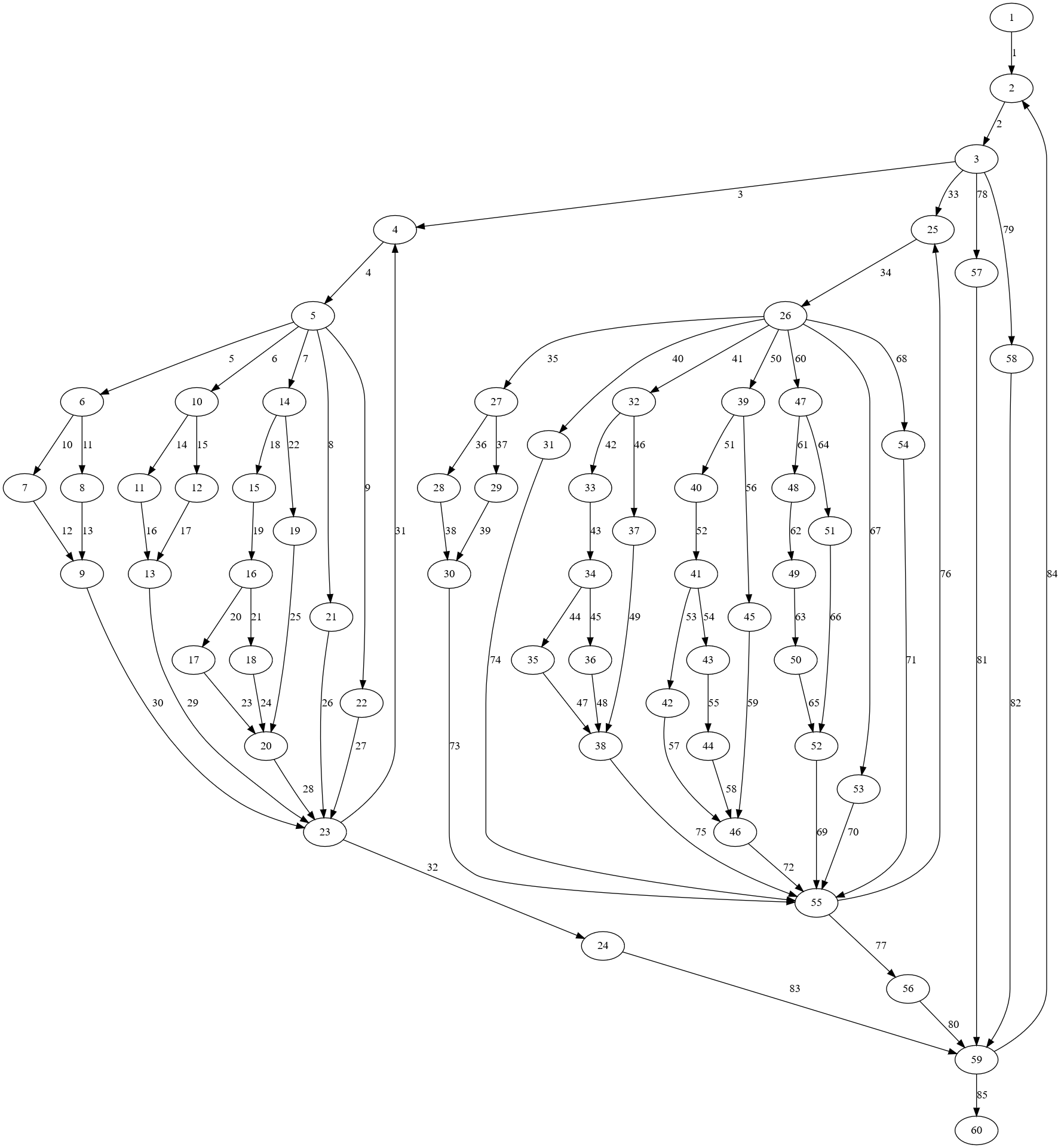


Рисунок 2 — Граф потока управления

Описание графа на языке DOT:

digraph GFG{

bgcolor = white

1->2 [label = "1"]

2->3 [label = "2"]

3->4 [label = "3"]

4->5 [label = "4"]

5->6 [label = "5"]

5->10 [label = "6"]

5->14 [label = "7"]

5->21 [label = "8"]

5->22 [label = "9"]

6->7 [label = "10"]

6->8 [label = "11"]

7->9[label = "12"]

8->9[label="13"]

10->11[label="14"]

10->12[label="15"]

11->13[label="16"]

12->13[label="17"]

14->15[label="18"]

15->16[label="19"]

16->17 [label="20"]

16->18 [label="21"]

14->19[label="22"]

17->20[label="23"]

18->20[label="24"]

19->20[label = "25"]

21->23[label="26"]

22->23[label = "27"]

20->23[label="28"]

13->23[label="29"]

9->23[label="30"]

23->4[label="31"]

23->24[label="32"]

3->25[label="33"]

25->26[label="34"]

26->27[label="35"]

27->28[label="36"]

27->29[label="37"]

28->30[label="38"]

29->30[label="39"]

26->31 [label="40"]

26->32 [label="41"]

32->33[label="42"]

33->34[label = "43"]

34->35[label="44"]

34->36 [label="45"]

32->37[label="46"]

35->38 [label = "47"]

36->38 [label = "48"]

37->38[label="49"]

26->39[label="50"]

39->40[label = "51"]

40->41[label = "52"]

41->42[label="53"]

41->43[label="54"]

43->44[label="55"]

39->45[label="56"]

42->46[label="57"]

44->46[label="58"]

45->46[label="59"]

26->47[label="60"]

47->48[label="61"]

48->49[label="62"]

49->50[label="63"]

47->51[label="64"]

50->52[label="65"]

51->52[label="66"]

26->53[label="67"]

26->54[label="68"]

52->55[label="69"]

53->55[label="70"]

54->55[label="71"]

46->55[label="72"]

30->55[label="73"]

31->55[label="74"]

38->55[label="75"]

55->25[label="76"]

55->56[label="77"]

3->57[label="78"]

3->58[label="79"]

56->59[label="80"]

57->59[label="81"]

58->59[label="82"]

24->59[label="83"]

59->2[label="84"]

59->60[label="85"]

}

Таблица 1 — Расчет метрики Маккейба:

|  |  |  |
| --- | --- | --- |
| Количество рёбер (дуг) | *m* | 85 |
| Количество узлов (вершин) | *n* | 60 |
| Цикломатическое число Маккейба | *M = m – n + 2* | 27 |

Для исчерпывающего тестирования программного кода (см. приложение А) потребуется 27 тестовых проходов, чтобы покрыть все пути исполнения.

Линейно-независимые маршруты и циклы для данного графа:

1)1->2->3->4->5->6->7->9->23->24->59->60

2)2->3->4->5->6->7->9->23->24->59->2

3) 2->3->25->26->27->28->30->55->56->59->2

4) 2->3->57->59->2

5) 2->3->58->59->2

6) 4->5->6->7->9->23->4

7) 4->5->6->8->9->23->4

8) 4->5->10->11->13->23->4

9) 4->5->10->12->13->23->4

10) 4->5->14->15->16->17->20->23->4

11) 4->5->14->15->16->18->20->23->4

12) 4->5->14->19->20->23->4

13) 4->5->21->23->4

14)4->5->22->23->4

15)25->26->27->28->30->55->25

16)25->26->27->29->30->55->25

17) 25->26->31->55->25

18) 25->26->32->33->34->35->38->55->25

19) 25->26->32->33->34->36->38->55->25

20) 25->26->32->37->38->55->25

21) 25->26->39->40->41->42->46->55->25

22) 25->26->39->40->41->43->44->46->55->25

23) 25->26->39->45->46->55->25

24) 25->26->47->48->49->50->52->55->25

25) 25->26->47->51->52->55->25

26) 25->26->53->55->25

27) 25->26->54->55->25

Таким образом, количество линейно-независимых маршрутов и циклов равно цикломатическому числу, что свидетельствует о корректно выполненном расчёте.

Выводы по работе

В результате выполнения данной работы представлена блок-схема алгоритма рассматриваемого программного кода, по которой был сформирован граф потока управления и выделены линейно-независимые маршруты и циклы. На основе графа потока управления выполнен расчет цикломатического числа Маккейба. При этом количество выделенных линейно-независимых маршрутов и циклов совпадает с цикломатическим числом, соответственно оценка цикломатической сложности программного кода выполнена корректно.

Для данного программного кода цикломатическое число Маккейба равно 27. Таким образом, чтобы выполнить исчерпывающее тестирование данного кода по принципу «работает каждая ветвь» потребуется 27 тестов. Однако это не означает, что программный код будет полностью протестирован, так как тестирование программного обеспечения в большей степени комбинаторная задача, и любая из технологий тестирования не позволяет отловить все ошибки программного кода, так как невозможно выполнить трассировку всевозможных путей выполнения программы, за исключением простейших случаев.

По отношению к рассматриваемому программному коду можно говорить о низкой алгоритмической сложности и соответственно высоких показателях свойств анализируемости и тестируемости. Высокая оценка данных характеристик основана на простой функциональности анализируемого кода и не связана с особенностями реализации.

Таким образом, можно заключить, что выполненная работа соответствует поставленной задаче и отвечает всем сформулированным в методических указаниях требованиям.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Павлов Е. В. Методические рекомендации к выполнению лабораторных работ: Метрология программного обеспечения / Евгений Васильевич Павлов. —   
   СПб ГУАП, 2021
2. Черников, Б. В. Управление качеством программного обеспечения: учебник /   
   Б. В. Черников. — М.: ИД «ФОРУМ»: ИНФРА-М, 2012. — 240 с.: ил.
3. Широков, А. И. Стандартизация, сертификация и оценка качества программного обеспечения: учебное пособие / А. И. Широков, Е. П. Потоцкий. —   
   М.: ИД «МИСиС», 2013. — 208 с.
4. Graph Description Language [Электронный ресурс]: Documentation / Emden R. Gansner, Eleftherios Koutsofios, Stephen North. — 2021. — URL: <https://graphviz.gitlab.io/_pages/pdf/dotguide.pdf> (дата обращения: 27.03.2021).

ПРИЛОЖЕНИЕ А

Анализируемый программный код

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199  200  201  202 | int main()  {  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  setlocale(LC\_ALL, "Russian");  system("color F1 ");  Dlist list;  Check check;  string resort, apart, food, code;  unsigned short int h, price, x = 0, z = 0, y = 0, l;  bool result;  savetimeanddate();  list.load();  do  {  cout << "\n\tГлавное меню:\n";  cout << "1. Поиск тура;\n2. Панель мастера обновления ПО;\n0. Выход из программы;";  x = check.checkc();  system("cls");  switch (x)  {  case 1:  do {  cout << "\n\tМеню:\n";  cout << "1. Вывести список всех предложений;\n2. Сортировать туры;\n3. Найти тур;\n0. Перейти в главное меню;";  z = check.checkc();  system("cls");  switch (z) {  case 1:  if (list.free()) {  cout << "Купить тур можно в нашем агенстве по адресу г. Санкт-Петербург ул. Цветочная д. 2 \nПо всем вопросам звоните на горячую линию: 8(800)555-35-35;\n";  list.show();  }  else {  cout << "На данный момент туров по стране нет\n";  }  break;  case 2:  if (list.free())  {  cout << "Что вы хотите сделать?\n\n";  cout << "1. Сортировать по возрастанию цены;\n2. Сортировать по убыванию цены;\n0. Перейти в меню;";  l = check.checkf();  list.sorting(l);  list.save();  }  else  {  cout << "Список предложений пуст!\n\n";  }  l = 0;  break;  case 3:  if (list.free())  {  cout << "\n1. Найти предложение по индивидуальному коду;\n" << "2. Найти элемент по названию курорта;\n";  l = check.checkf();  system("cls");  if (l == 1) {  cout << "Введите индивидуальный код предложения:\n";  cin >> code;  system("cls");  cout << "\nКупить тур можно в нашем агенстве по адресу г. Санкт-Петербург ул. Цветочная д. 2 \nПо всем вопросам звоните на горячую линию: 8(800)555-35-35;\n\n";  cout << "\nПо вашему запросу найдено:\n";  list.findelem(l, code);  }  else {  cout << "Введите название курорта:\n";  resort = check.checkname();  system("cls");  cout << "\nКупить тур можно в нашем агенстве по адресу г. Санкт-Петербург ул. Цветочная д. 2 \nПо всем вопросам звоните на горячую линию: 8(800)555-35-35;\n\n";  cout << "\nПо вашему запросу найдено:\n";  list.findelem(l, resort);  }  }  else  {  cout << "Список предложений пуст!\n\n";  }  l = 0;  break;  case 0:  break;  default:  break;  }  } while (z);  break;  case 2:  do {  cout << "\n\tМеню:\n";  cout << "1. Вывести список всех предложений;\n2. Добавить предложение;\n3. Изменить предложение;\n4. Удалить предложение;\n5. Удалить все предложения;\n0. Перейти в главное меню;";  y = check.checkc();  system("cls");  switch (y) {  case 1:  if (list.free()) {  list.show();  }  else {  cout << "Список предложений пуст!Заполните его.\n";  }  break;  case 2:  code = list.forcode();  check.fillandcheckdate(date);  cout << "Введите название курорта: ";  resort = check.checkname();  cout << "Введите тип и название апартаментов: ";  apart = check.checkname();  cout << "Введите кол-во человек: ";  h = check.chec();  cout << "Введите информацию о питании: ";  food = check.checkname();  cout << "Введите стоимость: ";  price = check.checkprice();  list.add(code, date, resort, apart, h, food, price);  cout << "Информация успешно добавлена.\n\n";  list.del();  list.save();  break;  case 3:  if (list.free())  {  cout << "Введите индивидуальный код предложения:\n";  cin >> code;  list.findelem(1, code);  result = list.editing(code);  if (result)  {  list.del();  list.save();  system("cls");  cout << "\nПредложение успешно отредактировано!\n\n";  }  else  {  system("cls");  cout << "\nТакого предложения нет!\n";  }  }  else  {  cout << "Список предложений пуст!Заполните его.\n\n";  }  break;  case 4:  if (list.free())  {  cout << "Введите индивидуальный код предложения:\n";  cin >> code;  list.findelem(1, code);  l = list.remove(code);  if (l == 1)  {  system("cls");  cout << "\nПредложение успешно удалёно!\n\n";  list.del();  list.save();  }  else if (l == 2)  {  cout << "\nТакого предложения нет!\n\n";  }  }  else  {  cout << "Список предложений пуст!\n\n";  }  l = 0;  break;  case 5:  if (list.free())  {  cout << "Вы точно хотите удалить все предложения?\nПосле удаления все текущие данные будут потеряны!\n\n";  cout << "1. Да;\n2. Нет;\n";  l = check.checkf();  if (l == 1)  {  list.clear();  list.del();  system("cls");  cout << "Данные удалены\n\n";  }  }  else  {  cout << "Список предложений пуст!\n\n";  }  break;  case 0:  break;  default:  break;  }  } while (y);  break;  case 0:  break;  default:  break;  }  } while (x); |
|  |  |